DOCUMENT RESUME

ED 382 180 IR 017 117

AUTHOR Bell, Benjamin; Korcuska, Michael
TITLE The Goal-Based Scenario Builder: Experiences with Novice Instructional Designers.
INSTITUTION Northwestern Univ., Evanston, IL. Inst. for the Learning Sciences.
PUB DATE 95
NOTE 14p.; Paper presented at the Annual Meeting of the American Educational Research Association (San Francisco, CA, April 1995).
PUB TYPE Speeches/Conference Papers (150) -- Reports - Research/Technical (143)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Authoring Aids (Programming); *Computer Assisted Design; Computer Assisted Instruction; Computer Interfaces; *Computer Software Development; *Courseware; Models; Multimedia Instruction; Tables (Data)
IDENTIFIERS Examples; Goal Based Education; *Interactive Systems

ABSTRACT

Creating educational software generally requires a great deal of computer expertise, and as a result, educators lacking such knowledge have largely been excluded from the design process. Recently, researchers have been designing tools for automating some aspects of building instructional applications. These tools typically aim for generality, resulting in interfaces with limited, generic styles. An alternative, appropriate for more complex, interactive software, is to provide tools with special purpose task models. A prototype authoring tool for interactive educational software, called Goal-Based Scenario Builder, is illustrated, and a mode of interaction called Guided Case Adaptation is described. Goal-Based Scenarios (GBS) provides an engaging task through which learners can master a set of target skills. This case-based approach to software design involves the user, and creates an interactive dialogue in which the program and designer collaboratively apply adaptations to a retrieved case. The Investigate and Decide GBS model is applied, and its five phases are outlined: problem, do, decide, communicate, and wrap-up. An example dialogue is presented. An early prototype of this tool was tested by graduate students in a first-year seminar in creating their own GBSs. An informal evaluation of the tool based on student reactions provided early indications that the tool makes prototyping easier, and that it supports a fairly wide range of applications within the limits set by the model. A table summarizes GBSs created in the prototype testing, and two figures illustrate the application. (Contains 15 references.) (MAS)

The Goal-Based Scenario Builder: Experiences with Novice Instructional Designers[1]

Benjamin Bell
bell@ils.nwu.edu

Michael Korcuska
korcuska@ils.nwu.edu

The Institute for the Learning Sciences
Northwestern University
Evanston, IL 60201

Abstract

Creating educational software generally requires a great deal of computer expertise, and as a result, educators lacking such knowledge have largely been excluded from the design process. Recently, researchers have been designing tools for automating some aspects of building instructional applications. These tools typically aim for *generality*, resulting in interfaces with limited, generic styles. An alternative, appropriate for more complex interactive software, is to provide tools with *special-purpose*, rather than general, task models. We illustrate a prototype authoring tool for interactive educational software, called Goal-Based Scenario Builder, and describe a mode of interaction called Guided Case Adaptation. We present an example dialog, and summarize a pilot evaluation of the tool.

keywords: Authoring systems, Interface design, Intelligent multimedia systems, Graphical user interfaces.

Introduction and Motivation

Building interactive software can be an arduous task, requiring expertise in both an application's domain and its user interface. Recently, researchers have been designing tools for automating some aspects of building interactive applications. These tools typically aim for *generality*, attempting to cover a broad class of interactive programs in many domains by relying on weak models, resulting in software whose interface has a limited, generic style.

This approach may not be effective for building complex interactive systems. An alternative approach is to provide tools which use *special-purpose*, rather than general, task models tailored specifically to the target software, which may have complex and

[1]Paper presented at the annual meeting of the American Educational Research Association, San Francisco, CA, April, 1995.

idiosyncratic interfaces. Moreover, in interacting with such a model, a designer should have access to example instantiations of the model (*i.e.*, other software programs created from that model).

The remainder of this paper presents a prototype tool, called *Goal-Based Scenario Builder*, for supporting the design of a specific type of instructional program, and summarizes a pilot evaluation of this prototype tool.

Goal-Based Scenarios and the paradigm of Investigation

People learn something because it helps them achieve some goal. Goal-Based Scenarios (Schank, 1994) provide a framework for computer-based instruction in which the learner is engaged in pursuing a goal, within a simulated environment, in order to master a set of target skills. The central tenet underlying the Goal-Based Scenario (GBS) framework is that skills are best learned when embedded within an engaging task, a position shared with other situated approaches to instructional technology (*e.g.* Bransford et al., 1990; Brown, Collins, & Duguid, 1989; CTGV, 1990). Goal-Based Scenarios also represent a methodology for the design of such programs (Schank, Fano, Jona, & Bell, 1994), so that any program created within this framework will be a piece of software which observes important design conventions.

An example of a GBS is SICKLE CELL COUNSELOR (Bell, Bareiss, & Beckwith, 1994), an exploratory hypermedia system designed for the Museum of Science and Industry in Chicago. This program provides the museum visitor with a basic understanding of genetics, and in particular, of Sickle Cell Disease, by allowing him or her to play the role of a genetic counselor, assisting clients who are considering having children, but are worried about the risk of Sickle Cell Disease. We performed three pilot evaluations of Sickle Cell Counselor, the results of which are described in (Bell, Bareiss, & Beckwith, 1994). The data from those evaluations suggest that the underlying architecture could help frame effective instruction in other domains.

Articulating this architecture resulted in a specialization of the GBS framework called *Investigate and Decide*. The name of this model captures its investigative nature - *e.g.*, in Sickle Cell Counselor, performing lab tests, asking questions of experts, and so on - and also highlights the reason behind the investigation - some central decision which the student is faced with, which requires some supporting knowledge the student must acquire.

The Investigate and Decide GBS model specifies five phases. In the **Problem** phase, students are introduced to the role they will play in the simulation. In Sickle Cell Counselor (SCC), for example, users are told that they will be acting in the role of genetic counselors. The **Do** phase is where the principal investigation activities associated with the

task take place, which in the case of SCC includes viewing blood samples and performing an electrophoresis test to establish the clients' genotypes. The **Decide** phase is where students interpret data gathered in the Do phase in order to reach a decision (*e.g.*, by using the Punnett Square in SCC). In the **Communicate** phase the student conveys that decision to agents in the program, as in the case of SCC where the user advises the clients. The **Wrap-Up** phase offers some closure to the interaction by demonstrating the effects of the student's final decision; in the case in SCC, the clients return "a year later" to talk about the ultimate outcomes.

The utility of articulating this model lies in its potential as a blueprint for new GBSs. A specialized design tool built around such a model stands in contrast to more general approaches described below.

Approaches to software design tools

General-purpose authoring tools such as Hypercard and Authorware are intended to facilitate the mechanics of design, but not the more critical work of constructing a coherent, effective program. A better tool might allow the designer to create individual components of a course, supplying an abstract sketch of a complete program (*e.g.* Murray & Woolf, 1992). Better still would be a tool which would offer examples of successful solutions to design problems, and would provide sufficient guidance to ensure that the resulting course is pedagogically sound. In other words, the tool should be general-purpose, but only to a point: the tool must support a user in creating a program which is properly conceived.

General-purpose tools vary in scope and structure. Two classes relevant to this discussion are interface generation tools and authoring environments for intelligent tutoring systems (ITSs). Since creating a user interface tailored to an application can be a programming-intensive task, an aim of interface generation tools is to automate (entirely or partially) the interface construction process. Such tools share the goal of creating custom interfaces from parameters supplied to a model by the designer. A number of model-based interface generation systems have been reported in the literature, including HUMANOID (Szekely, Luo, & Neches, 1993), MECANO (Puerta, Eriksson, Gennari, & Musen, 1994), and GENIUS (Janssen, Weisbecker, & Ziegler, 1993). Although these tools differ in many respects, they each attempt to generate an interface based on a declarative description of the desired presentation and behavior of the interface. Each of these approaches aims for *generality*, attempting to cover a broad class of interactive programs in many domains. One problem is that these tools rely on generic models of interfaces (usually form- or menu-based). A consequence of this approach is that the range of tasks

these tools can support is limited (the above examples support the design of browsing and editing applications, for example).

A second category of tools supports the creation of Intelligent Tutoring Systems (*e.g.*, Macmillan, Emme, & Berkowitz, 1988; Murray & Woolf, 1992; Russell, Moran, & Jordan, 1988). Tools in this category are somewhat specialized in that they are restricted to designs for instructional systems, however, they still aim to support the creation of any possible kind of instruction. In doing so, these tools fall short of being truly useful, general-purpose tools for several reasons. First, they lack the knowledge required to dispense design advice about the structure of an end-user's activities, or about the elements which would populate these activities. Second, these tools provide generic (and therefore limited) interfaces, so as to apply to a wide range of tasks, since the alternative is to include a large catalog of interfaces, which introduces complexities associated with indexing a large collection of complex artifacts and with generating appropriate retrieval strategies. Third, the design guidance these tools provide is in terms which are general enough to accommodate any program; in other words, some of the tool's value as a knowledge-based design aid is sacrificed in pursuit of generality.

We adopt an alternative approach by providing a tool which uses a special-purpose, rather than a general, task model. To augment the model, we introduce an exemplar which works in concert with the model to guide the design process. This mode of interaction, called *Guided Case Adaptation*, is described in the following section.

Guided case adaptation as a GBS design method

Case-Based design research, specifically, work in case adaptation, has focused primarily on automating the adaptation process (*e.g.*, Hinrichs & Kolodner, 1991; Kass, 1989). An alternative is to involve the user, creating an interactive dialog in which the program and designer collaboratively apply adaptations to a retrieved case. This approach is appropriate for domains in which cases are likely to be quite complex and design knowledge difficult to codify. Moreover, when a model can be leveraged to systematize such a dialog , the process becomes one which can be termed *Guided Case Adaptation.*

It is the dynamic among the designer, model, and exemplar, which characterizes this mode of interaction. While an instructional designer could, in theory, instantiate the abstract design model directly to create a GBS, we believe this would be extremely difficult in contrast to adapting a specific instance of that model. On the other hand, if the designer were confronted simply with an artifact such as Sickle Cell Counselor, he or she would lack sufficient guidance as to which aspects of the program are likely to be relevant and

adaptable. Our approach, therefore, combines a specific example (Sickle Cell Counselor) with an abstract model of GBSs to guide the designer's adaptation process.

This mode of interaction is well-suited the design of GBSs for two reasons: First, it does not require a system to possess deep, operationalized knowledge of design goals and of which specific design choices best satisfy them. Instead, such design choices are made by the user, aided by an example GBS augmented with design rationale. Second, the user is able to observe the adaptation process incrementally as each alteration is applied. This enables a user to make each decision in a logical sequence, observing its effects on the emerging artifact and its consistency with prior design choices.

What still remains to be seen, though, is how a program can convey to a designer the twin notions of, on the one hand, the abstract model of Investigate and Decide GBSs, and on the other hand, a specific exemplar from this class. The program which does this is called *GBS Builder* and is the focus of the next section.

A Case-Based Tool for Constructing Investigate and Decide GBSs

GBS Builder is a prototype implementation of the guided case adaptation approach. It is intended for use by a domain expert or teacher. To illustrate the way a user interacts with GBS Builder, we present an abridged sample dialog (a more detailed example appears in Bell, Kedar, and Bareiss, 1994). To set the stage for the session, consider the designer to be a high school chemistry teacher, who wishes to teach students about the molecular makeup of various compounds by engaging them in the role of Arson Investigator. In the emerging GBS, the student seeks to determine whether or not a fire was set intentionally, by looking for evidence of accelerant chemicals in samples taken at the scene.

Getting started: the Problem phase

The program elicits input from the designer one phase at a time, as specified by the model. The interaction during each phase proceeds first by previewing the elements the designer will be asked to specify, then by assisting the designer in the actual adaptation. After a phase is introduced, each subphase is processed in sequence. The first such subphase in the example is introduced as follows:

In the ATTRACT USER STAGE, you will specify:

- a title for the GBS
- a prompt for how to begin
- one or more movies introducing problems the student can choose from

To define, *e.g.*, the prompt, the designer types into a text window, as shown in Figure 1.

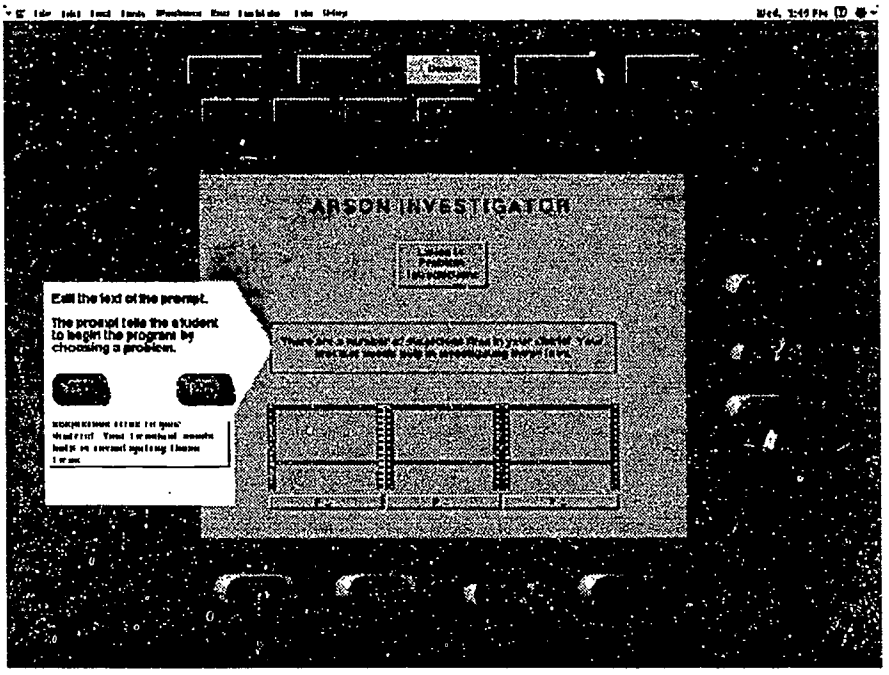

Figure 1. Specifying a startup prompt

Defining an activity screen: the Do phase

During the Do phase, students acquire information on which to base the decision posed in the GBS. Although the Do phase is more complex than the Problem phase, the model still guides the designer through what amounts to adaptation. An abstract model of investigation defines the relationships among elements and how the end-user interacts with them.

The designer need only provide labels and pictures for instantiating this abstraction as a specific investigation. The model defines a typical investigation in this class of GBSs as including steps for acquiring a sample from a specified source, selecting a test to be applied to that sample, performing the test, and obtaining the test results. In the Arson Investigator example, the designer wishes to specify an investigation in which the student extracts a sample of drapery fibers and performs a chromatography test to determine whether the sample contains accelerants. Figure 2 illustrates the designer specifying a video to depict a sample being collected.

This brief example illustrates that using the tool to create a new GBS is *possible*. To begin understanding whether the tool was *useful*, we conducted an informal study of graduate students using this tool, which we discuss in the following section.

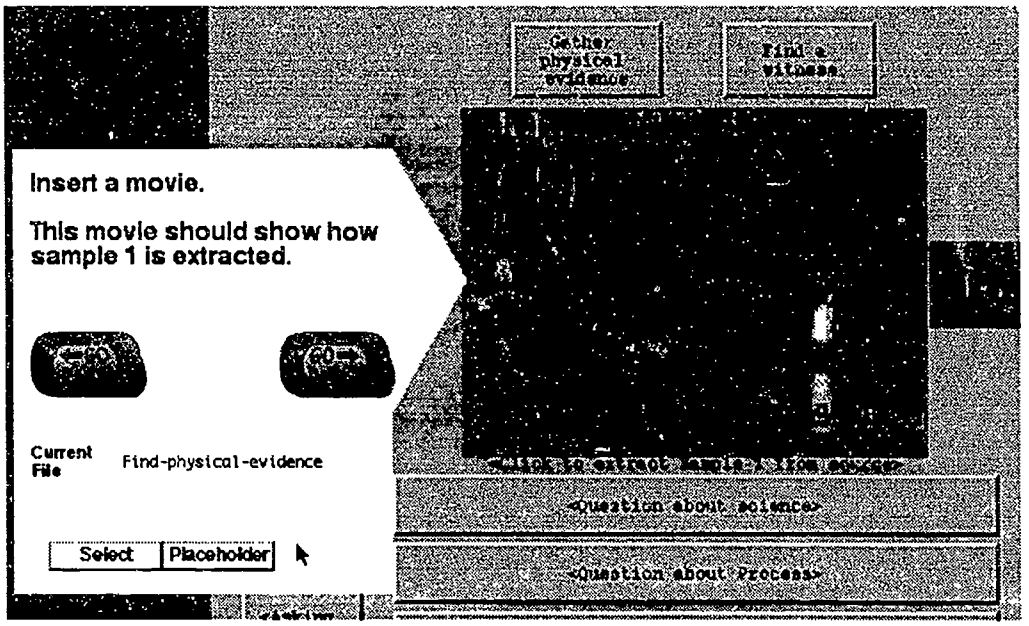

Figure 2. (Detail) Specifying how sample extraction appears in the Do phase

A Pilot Evaluation of GBS Builder

The GBS Builder is intended for a particular audience (instructional designers and teachers), and for a particular purpose (the construction of Investigate and Decide GBSs). We therefore wanted to test the validity of a number of assumptions we were making about our users, and to discover whether some design changes we were considering would, in fact, be beneficial. To help answer these questions we organized a pilot study of GBS Builder, in which first-year graduate students participating in a seminar were assigned the task of creating GBSs using the tool. The aim of this pilot study was to determine how valid our basic assumptions were and to gather formative data in support of on-going development of the tool. The questions addressed by this preliminary study were:

- What range of ideas can be tailored to fit the tool?
- Do highly specific tools encourage the production of better software?
- Does the tool effectively support the prototyping process?
- Does the tool embody a level of specificity that designers find comfortable?
- Is the guidance provided by the tool appropriate for the intended audience?

Description of the Experiment

Twenty-one students enrolled in the seminar (19 were first-year graduate students and 2 were undergraduate seniors). They were shown demos of SCC and of GBS Builder. They then paired into 10 teams and each was asked to design and implement a GBS, using GBS Builder. After three weeks of design work, each team's preliminary design was critiqued by the faculty member running the seminar. After three more weeks, each team showed its final product. As part of the required work, each team prepared a short paper summarizing their experiences in building a GBS and in using the tool.

Results

The teams created ten GBSs using the GBS Tool. The resulting GBSs encompassed a wide variety of domains and student missions, which indicates that the tool can accommodate a fairly broad range of ideas. This variety is, however, somewhat misleading. Because the authors were experienced computer programmers (in some cases at least) they were able to circumvent some of the tool's specificity and create programs that were not strictly Investigate and Decide GBSs. Since we intended the tool to be able to create only Investigate and Decide GBSs, we need to take these departures into account in order to accurately judge how wide a range of domains the tool supports. To do this we classified the programs constructed by the authors into three groups: those GBSs which closely followed the structure of Sickle Cell Counselor (classified as *investigate and decide*), those which preserved the browsing aspect of SCC but omitted the testing element (classified as *browsing systems*), and those which provided a simple simulation in lieu of running tests (classified as *simulation* GBSs). Summary descriptions of the GBSs appear in Table 1. GBSs in the Investigate and Decide category represent programs that the tool is intended to support. The other two categories contain programs which are meaningful departures from what the tool is intended to build.

As the table indicates, there was a wide variety of domains even within the Investigate and Decide category. This sheds positive light on the question of how wide a range of ideas the tool could support. It is difficult to say whether or not the tool helped the authors create effective educational software. Formal evaluations of these programs were not performed, since creating them took the entire length of the seminar. We did perform a limited evaluation, in order to assign grades for the seminar. The programs on the whole were interesting to use and presented coherent scenarios. A problem shared by the GBSs was a vague notion of the instructional goals, although one pattern which emerged was that the GBSs which remained closest to the Investigate and Decide model had the clearest

instructional goals, and those which ventured furthest from the model were interesting as games but did not seem to be teaching much beyond how to "win". This provides some preliminary support for our belief that the constraints imposed by a highly specialized tool will help designers produce educationally effective software.

Table 1. GBSs created using GBS Builder

GBS Type	Title	Mission
Investigate & Decide	Aztec, P.I.	advise curator about objects' authenticity
	Cardiac Counselor	advise clients at risk for heart disease
	Environmental Decision Maker	take action at a polluted site
	Public Health Advisor	advise mayor about outbreak of illness
	Secret Caves of Ellora	find solution to archeological problem
simulation	Crash Course	determine driver at fault in traffic accident
	Diver Down	help plan scuba dives
browsing system	GBSex	counsel clients about contraception
	Rags to Riches	advise investors
	Scandal!	draft news story about a scandal

In addition to creating their GBS, each team was asked to record their reactions to the tool, and to offer suggestions about improving its design. Ten papers were submitted describing each team's reaction to the tool and suggestions for extending the tool. In general, teams reported that they liked the basic structure of the tool and that using it yielded a speed-up during the initial design phase. This helped answer the question of whether the tool made the prototyping process easier.

The teams also reported suggestions for changing things which they did not like about the tool. The most common suggestion (60%) was to make it easier to build programs that did not follow the Investigate and Decide instructional model. This is reflected in some of their comments:

> "It would get very confusing if you could not directly map the elements of your design to Sickle Cell."
>
> "Ease of Use for mapping directly to Sickle Cell: 5; Ease of Use for any deviation from the Sickle Cell Design: 1."

As a solution to this problem, several teams proposed making the tool's internal data structures more open to modification. We attribute this reaction to the makeup of the subject group. That is, the team members were experienced programmers, and, we believe, eager to display their creative talents even when (or perhaps especially when) this meant operating beyond the intended functionality of the tool. Since our true intended audience are educators and instructional designers, and since our aim is provide a tool which maintains the boundaries implicit in the model, we do not regard the proposed solution as appropriate. More important is the problem the teams faced, namely, that the Investigate and Decide model was not appropriate for all their ideas. This problem emphasizes two important issues. First, it makes it clear that many initial designs will need to be tailored to fit the tool and that the tool should support this process. Second, it implies that tools to support instructional models other than Investigate and Decide will need to be created.

The next most common reaction (50%) was that the tool needed to provide guidance which was more conceptually-oriented and less interface-oriented. This was exactly the kind of guidance we suspected the tool would need but which we had not, at the time, been able to include. The tool used in the experiment guided authors through a step-by-step adaptation of the exemplar GBS, but did not attempt to explain the overarching concepts governing the individual steps. This interface guidance fell short of the kind of conceptual guidance the subjects said they wanted to see, with the reported consequences including losing focus on content design, and losing track of the overall flow because editing proceeds one screen at a time. The comments below are typical:

> "There is nothing in the tool which allows you to see the control flow of the GBS. Something like a flow chart would be useful."
> "Without an outline or some other a priori structure the tool can be confusing."

Implications for the GBS Builder Tool

Results gathered from the seminar suggest several issues regarding the tool, which can be broken down into three categories:

- interface construction
- alternative models
- task-level guidance

Interface construction

As a practical issue, students in the study reported that working through each item in the interview script was useful for the first pass, but became cumbersome as a way to "zoom in" and change particular parameters. Our proposed solution to this is to allow items on the screen to be selected for editing; items which do not have a corresponding screen element may be listed in a separate window for selection. The tool development team is implementing this and other interface improvements.

Alternative models

Of more theoretical interest is the desire on the part of the graduate students in this study to work "outside of the tool", that is, to design programs which did not necessarily observe the structure of the model. In fact, 40% of the design teams omitted portions of the model or reordered or combined steps internal to the model, and 60% of the teams added parts to the model.

One solution to alleviate this problem is to augment the templates by providing variant paths, *e.g.*, optionally skipping subphase-level steps, or reordering steps. Providing some local flexibility could allow the tool to accommodate a broader range of programs while preserving the intent of the model. We had recognized before the experiment that this kind of local flexibility would be a desirable feature for the tool to have, but it is difficult to anticipate what flexibility designers will actually want. One of the most important results of the experiment was, in fact, to clarify precisely which local variations would be most useful.

The approach which the subjects in this study supported, though, was to redesign the tool to accommodate a variety of models. A problem with this idea is that the tool supports the design of a *specific* class of GBSs. This tool is specialized, not only in the model, but in the various editors, interview scripts, and other support features particular to this class of GBS. Attempting to revise the entire tool structure to support each model would either result in too general a tool, or would amount to the approach we are advocating, namely, constructing special-purpose tools for each class of GBS.[2]

Task-level guidance

The most theoretically relevant reactions from the students related to the nature of the guidance provided by the GBS tool. The need for guidance arises from two sources. First, in order for a tool to be useful, it must assist a designer in the mechanics of using it.

---

[2]This is not to say that tools for different GBS types will not share common elements. We are currently defining the components each class of GBS will have, in order to isolate the shared components.

Second, because it is very much a special purpose tool, it must also make very clear to the designer what constraints it will be enforcing, why these constraints exist, and how the designer can observe them. The GBS tool used by the students in the seminar provided only the former kind of guidance. That is, the tool offered help regarding what should be done, and how, but not why.[3]

The students' reactions to the procedural emphasis of the guidance provided a strong indication that the tool should adopt a more global stance in guiding the design process. In particular, the data which the program asks the designer to provide should describe the emerging GBS in conceptual terms, not just in terms of its interface. This is the most significant result from this pilot study, and so the question becomes how to change the tool to provide conceptual guidance. We believe that encoding knowledge about the task into the tool will allow GBS Builder to offer more conceptual guidance. This work is ongoing in a number of projects at ILS.

Conclusion

The Investigate and Decide model introduced in this paper serves to package a set of design principles around which an authoring environment may be created. We presented GBS Builder, a tool for the creation of a special class of GBSs. The tool steps the designer through the process of adapting an exemplar GBS, using the model to guide the adaptation. We described this mode of interaction as guided case adaptation, drawing parallels to case-based design.

An early prototype of this tool was used by graduate students in a first-year seminar, in creating their own GBSs. An informal evaluation of the tool based on student reactions provided early indications that the tool makes prototyping easier, and that it supports a fairly wide range of applications within the limits set by the model. Getting an initial design to fit the model, though, was problematic in some cases, and examining these cases proved useful in rethinking some of the tool's design.

References

Bell, B.L., Bareiss, R., and Beckwith, R. (1994). Sickle Cell Counselor: A Prototype Goal-Based Scenario for Instruction in a Museum Environment. *Journal of the Learning Sciences, 3*(4), 347-386.

[3]Design rationale was provided by a "why?" button, which explained the role of a particular element of the GBS, but no higher-level justifications were available.

Bell, B.L., Kedar, S., and Bareiss, R. (1994). "Interactive Model-Driven Case Adaptation for Instructional Software Design". In *Proceedings of the Sixteenth Annual Meeting of the Cognitive Science Society,* Atlanta, GA, 1994.

Bransford, J.D., et al. (1990). Anchored instruction: why we need it and how technology can help. In R. Spiro and D. Nix (Eds.), *Cognition, Education, and Multimedia: Exploring Ideas in High Technology.* Lawrence Erlbaum: Hillsdale, N.J.

Brown, J.S., Collins, A., and Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18.*(1), 32–41.

Cognition and Technology Group at Vanderbilt (CTGV). (1990). Anchored Instruction and its relationship to Situated Cognition. *Educational Researcher, 19*(6), 2–10.

Hinrichs, T.R. and Kolodner, J.L. (1991). The Role of Adaptation in Case-Based Design. In *Proc. of the Case-Based Reasoning Workshop,* pp. 121-132, Washington, D.C.

Janssen, C., Weisbecker, A., and Ziegler, J. (1993). "Generating User Interfaces from Data Models and Dialogue Net Specifications." In *INTERCHI '93.*, (Amsterdam, April, 1993), ACM Press, pp.418-423.

Kass, A.M. (1989) *Developing Creative Hypotheses by Adapting Explanations.* PhD Thesis, Yale University. Issued as Technical Report #6, Institute for the Learning Sciences, Northwestern University, Evaston, IL.

Macmillan, S., Emme, D., and Berkowitz, M. (1988). *Instructional planners: Lessons learned.* In Psotka, Massey, and Mutter (Eds.), Intelligent Tutoring Systems, Lessons Learned. Hillsdale, NJ: Lawrence Erlbaum.

Murray, T. and Woolf, B.P. (1992). A knowledge acquisition tool for intelligent computer tutors. *SIGART Bulletin, 2*(2), 9-21.

Puerta, A.R., Eriksson, H., Gennari, J.H., and Musen, M.A. (1994). "Model-Based Automated Generation of User Interfaces". In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, 1994.

Russell, D., Moran, T.P., and Jordan, D.S. (1988). *The instructional design environment.* In Psotka, Massey, and Mutter (Eds.), Intelligent Tutoring Systems, Lessons Learned. Hillsdale, NJ: Lawrence Erlbaum.

Schank, R.C. (1994). "Goal Based Scenarios: A Radical Look at Education." *The Journal of the Learning Sciences*, *3*(4), 429-453

Schank, R.C., Fano, A., Bell, B.L. , and Jona, M.Y. (1994). "The Design of Goal Based Scenarios." *The Journal of the Learning Sciences*, *3*(4), 305-345.

Szekely, P., Luo, P., and Neches, R. (1993). "Beyond Interface Builders: Model-Based Interface Tools". In *INTERCHI '93*, (Amsterdam, April, 1993), ACM Press, pp.383-390.